US009311477B2

(12) **United States Patent**
Marinet et al.

(10) **Patent No.: US 9,311,477 B2**
(45) **Date of Patent: Apr. 12, 2016**

(54) **METHOD AND DEVICE FOR FAULT DETECTION**

(71) Applicants: **Proton World International N.V.**, Zaventem (BE); **STMicroelectronics (Rousset) SAS**, Rousset (FR)

(72) Inventors: **Fabrice Marinet**, Chateuneuf le Rouge (FR); **Jean-Louis Modave**, Ottignies (BE); **Gilles Van Assche**, Woluwe-St-Lambert (BE); **Ronny Van Keer**, Hoeilaart (BE)

(73) Assignees: **Proton World International N.V.**, Amsterdam (NL); **STMicroelectronics (Rousset) SAS**, Rousset (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 301 days.

(21) Appl. No.: **13/715,157**

(22) Filed: **Dec. 14, 2012**

(65) **Prior Publication Data**

US 2013/0159791 A1 Jun. 20, 2013

(30) **Foreign Application Priority Data**

Dec. 15, 2011 (FR) 11 61673

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 21/55 (2013.01)

(52) **U.S. Cl.**
CPC ***G06F 21/554*** (2013.01); *G06F 2207/7219* (2013.01)

(58) **Field of Classification Search**
CPC G06F 11/0709; G06F 11/0778; G06F 11/079; G06F 11/0793; G06F 11/3466; G06F 2207/7219; G06F 2207/7223
USPC 714/37, 48
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,111,126	B2	9/2006	Biles et al.
7,848,515	B2	12/2010	Dupaquis et al.

(Continued)

FOREIGN PATENT DOCUMENTS

FR	2841015	A1	12/2003
FR	2 919 739	A1	2/2009

OTHER PUBLICATIONS

Dictionary definition for “processor register” from wilipedia using the way back machine, retrieved from https://web.archive.org/web/20111107124250/http://en.wikipedia.org/wiki/Processor_register, from date Nov. 7, 2011.*
Bar El et al. “The Sorcerer’s Apprentice Guide to Fault Attacks”, Proceedings of the IEEE, vol. 94, No. 2, Feb. 2006.*
French Search Report, dated Jul. 13, 2012 for French application No. 1161673, 7 pages.

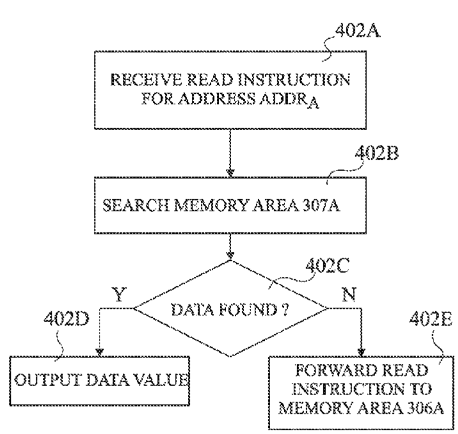
(Continued)

Primary Examiner — Yair Leibovich
(74) *Attorney, Agent, or Firm* — Seed IP Law Group PLLC

(57) **ABSTRACT**

The disclosure concerns a method implemented by a processing device. The method includes performing a first execution by the processing device of a computing function based on one or more initial parameters stored in a first memory device. The execution of the computing function generates one or more modified values of at least one of the initial parameters, wherein during the first execution the one or more initial parameters are read from the first memory device and the one or more modified values are stored in a second memory device. The method also includes performing a second execution by the processing device of the computing function based on the one or more initial parameters stored in the first memory device.

22 Claims, 7 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

7,856,523	B2	12/2010	Bittner, Jr.	
8,688,995	B2	4/2014	Teglia	
2005/0157871	A1	7/2005	Komano et al.	
2006/0045264	A1	3/2006	Kocher et al.	
2006/0212770	A1	9/2006	Fischer et al.	
2007/0019805	A1	1/2007	Karpovsky et al.	
2007/0177720	A1	8/2007	Bevan et al.	
2007/0188355	A1	8/2007	Baek	
2007/0286413	A1	12/2007	Derouet	
2009/0044265	A1*	2/2009	Ghosh et al.	726/14
2009/0323956	A1	12/2009	Tsunoo et al.	
2011/0041013	A1*	2/2011	Ingimundarson	714/40
2011/0119532	A1	5/2011	Teglia	
2011/0119762	A1*	5/2011	Teglia	726/23
2011/0225432	A1*	9/2011	Trichina	713/190
2013/0103972	A1*	4/2013	Ozer et al.	714/2
2013/0305098	A1*	11/2013	Keromytis	G06F 11/08 714/38.1

OTHER PUBLICATIONS

Bar-El et al., "The Sorcerer's Apprentice Guide to Fault Attacks," retrieved from http://web.archive.org/web/20041016071838/eprint.iacr.org/2004/100, Oct. 16, 2004, XP002329915, 13 pages.

Bertoni et al., "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard," *IEEE Transaction on Computers* 52(4):492-505, Apr. 2003.

Karpovsky et al., "Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard," Proceedings of the 2004 International Conference on Dependable Systems and Networks, Jun. 28-Jul. 1, 2004, Piscataway, NJ, 9 pages.

Karri et al., "Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21(12):1509-1517, Dec. 2002.

Moratelli, Carlos R. et al., "A Cryptography Core Tolerant to DFA Fault Attacks," Journal Integrated Circuits and Systems 2007; v.2, n. 1, pp. 14-21.

* cited by examiner

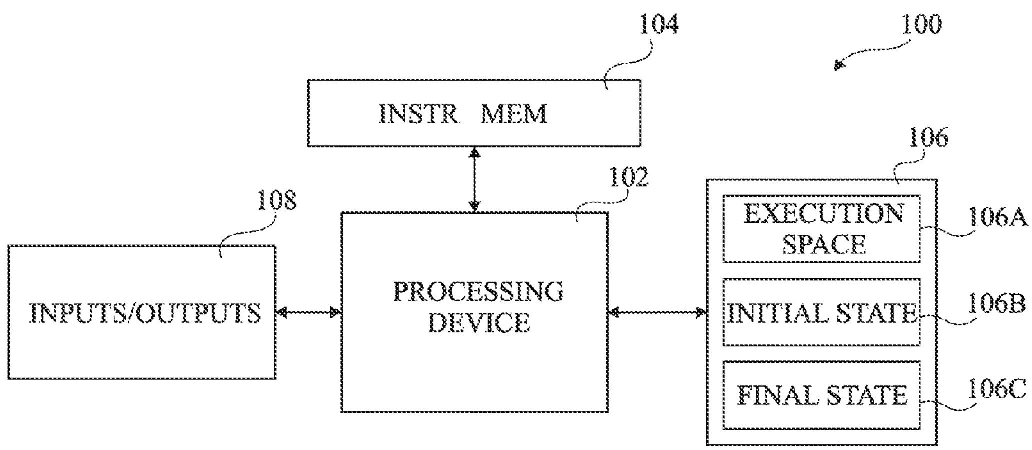

104
100
INSTR MEM
106
102
108
EXECUTION SPACE
106A
INPUTS/OUTPUTS
PROCESSING DEVICE
INITIAL STATE
106B
FINAL STATE
106C

Fig 1

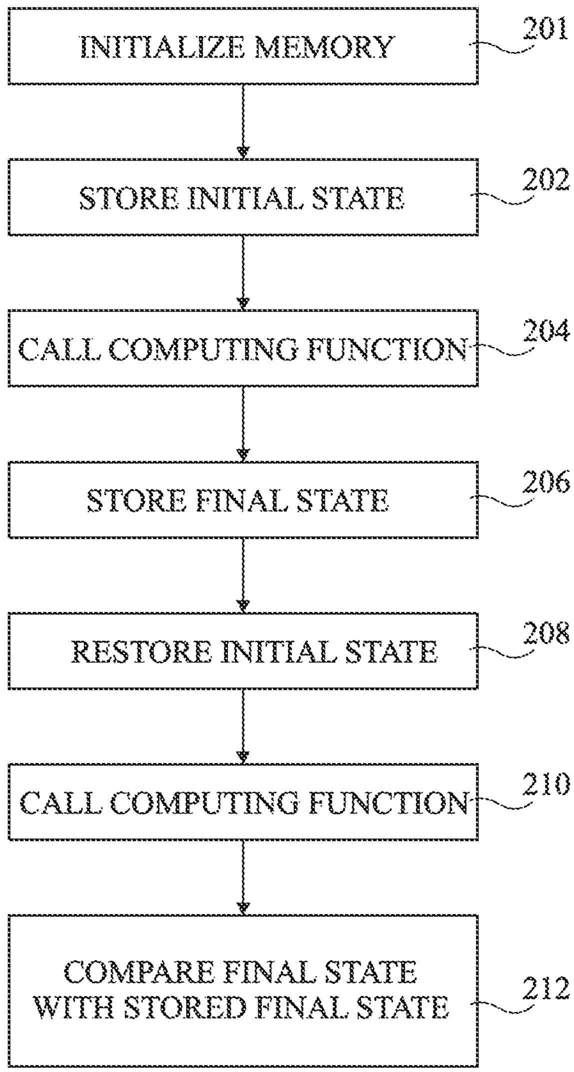

INITIALIZE MEMORY
201
STORE INITIAL STATE
202
CALL COMPUTING FUNCTION
204
STORE FINAL STATE
206
RESTORE INITIAL STATE
208
CALL COMPUTING FUNCTION
210
COMPARE FINAL STATE WITH STORED FINAL STATE
212

Fig 2

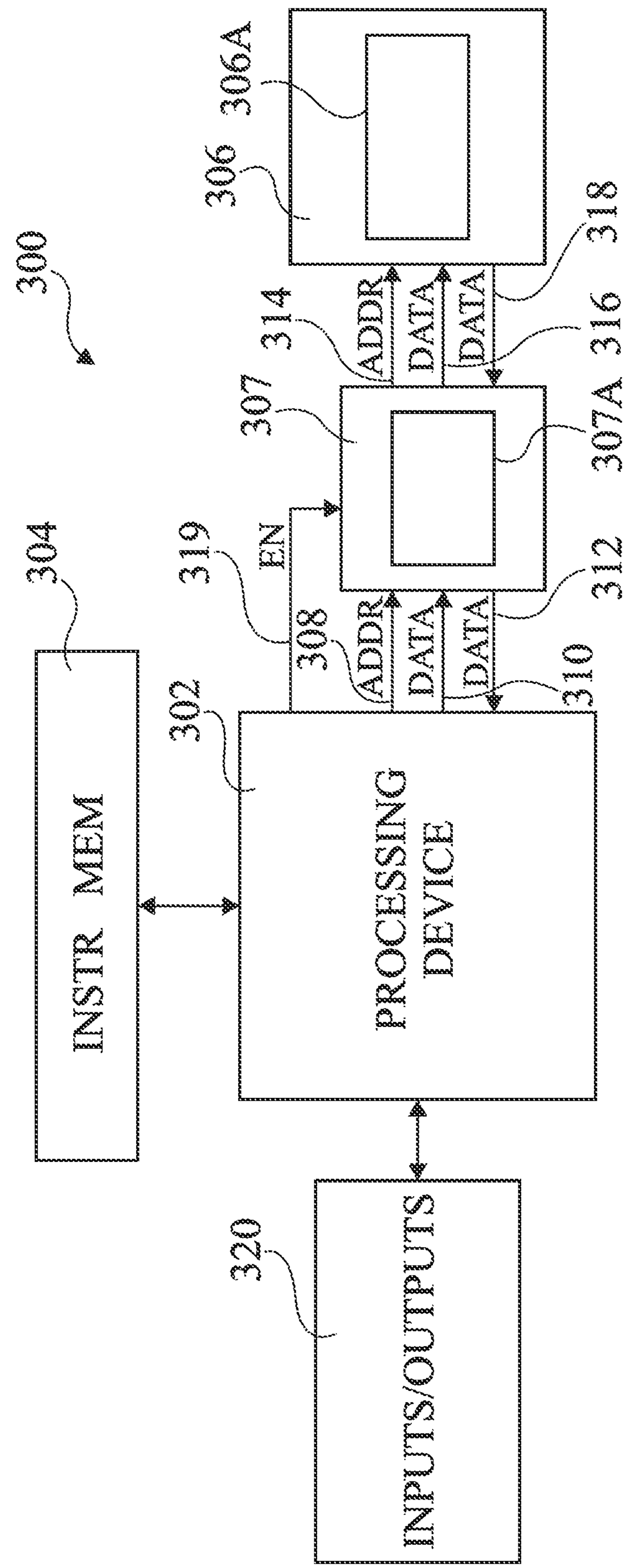

300
304
INSTR MEM
302
PROCESSING DEVICE
319
EN
308
ADDR
DATA
DATA
310
312
307
307A
314
ADDR
DATA
DATA
316
318
306
306A
320
INPUTS/OUTPUTS

Fig 3

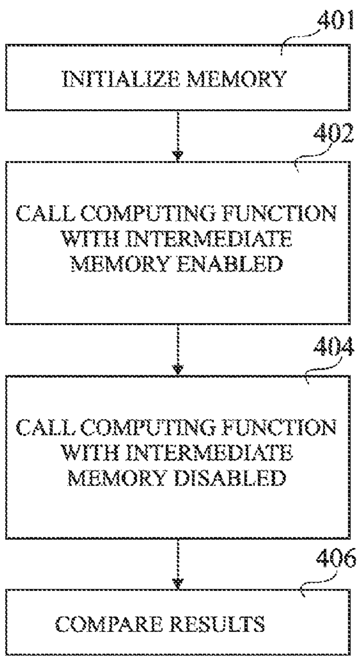

401
INITIALIZE MEMORY
402
CALL COMPUTING FUNCTION WITH INTERMEDIATE MEMORY ENABLED
404
CALL COMPUTING FUNCTION WITH INTERMEDIATE MEMORY DISABLED
406
COMPARE RESULTS

Fig 4A

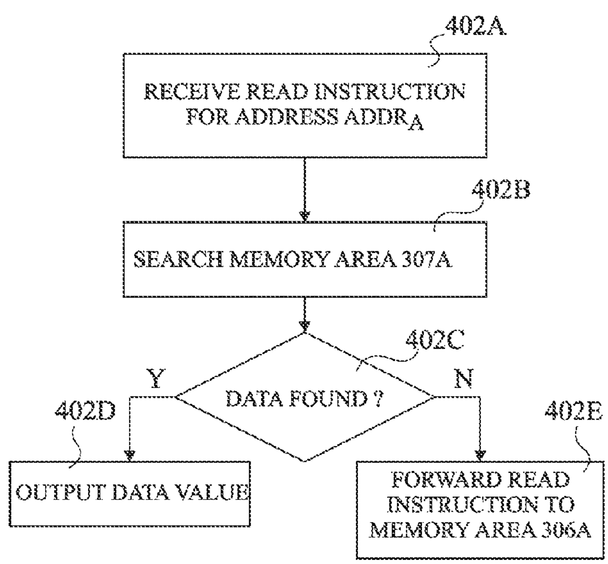

402A
RECEIVE READ INSTRUCTION FOR ADDRESS ADDR_A
402B
SEARCH MEMORY AREA 307A
402C
DATA FOUND ?
Y
N
402D
OUTPUT DATA VALUE
402E
FORWARD READ INSTRUCTION TO MEMORY AREA 306A

Fig 4B

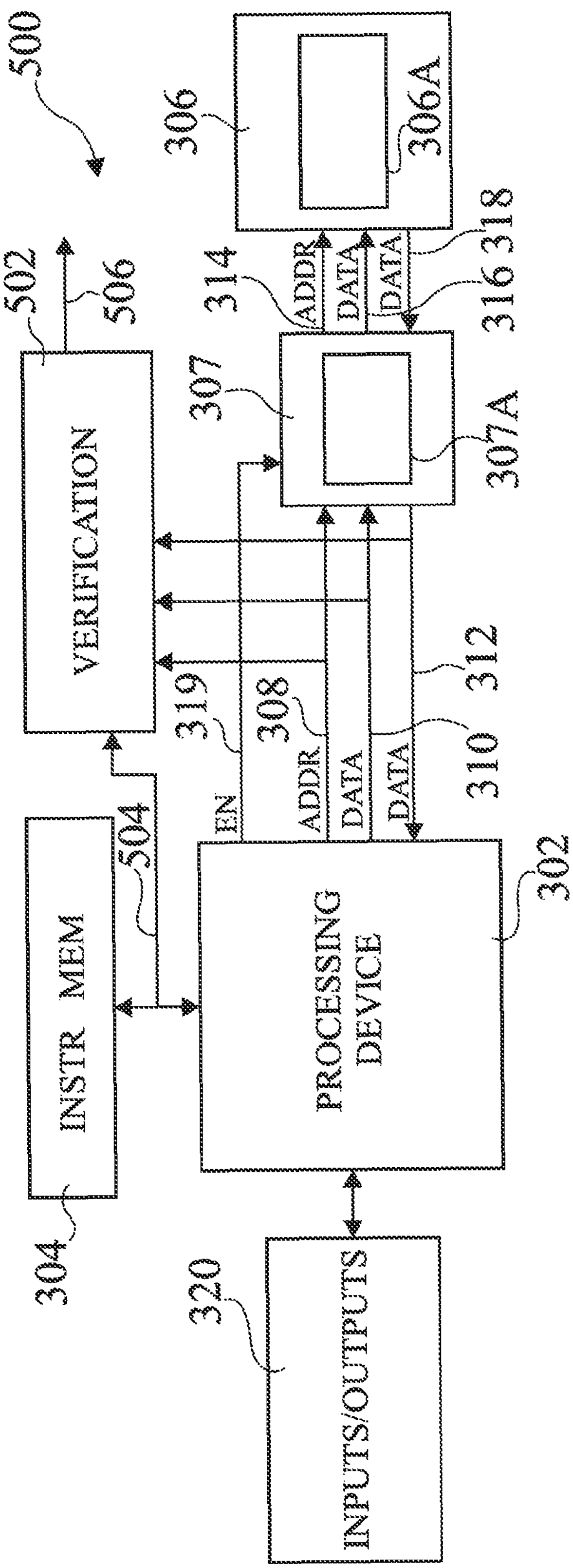

500
306
306A
502
506
314
318
316
ADDR
DATA
DATA
307
307A
VERIFICATION
319
308
310
312
EN
ADDR
DATA
DATA
504
INSTR MEM
PROCESSING DEVICE
302
304
320
INPUTS/OUTPUTS

Fig 5

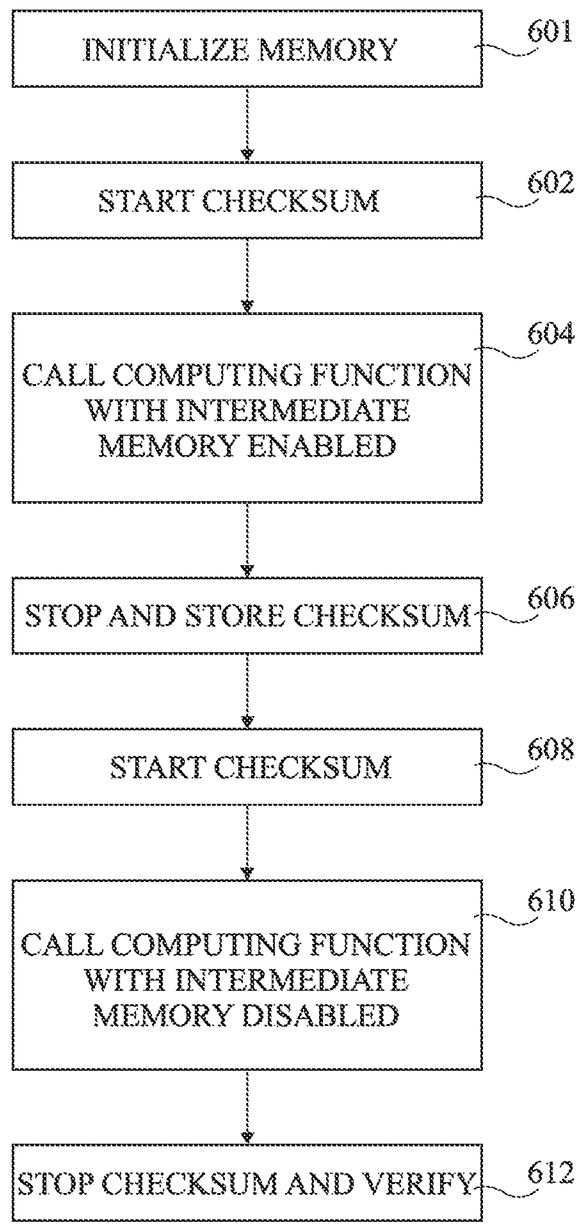

INITIALIZE MEMORY
601
START CHECKSUM
602
CALL COMPUTING FUNCTION WITH INTERMEDIATE MEMORY ENABLED
604
STOP AND STORE CHECKSUM
606
START CHECKSUM
608
CALL COMPUTING FUNCTION WITH INTERMEDIATE MEMORY DISABLED
610
STOP CHECKSUM AND VERIFY
612

Fig 6

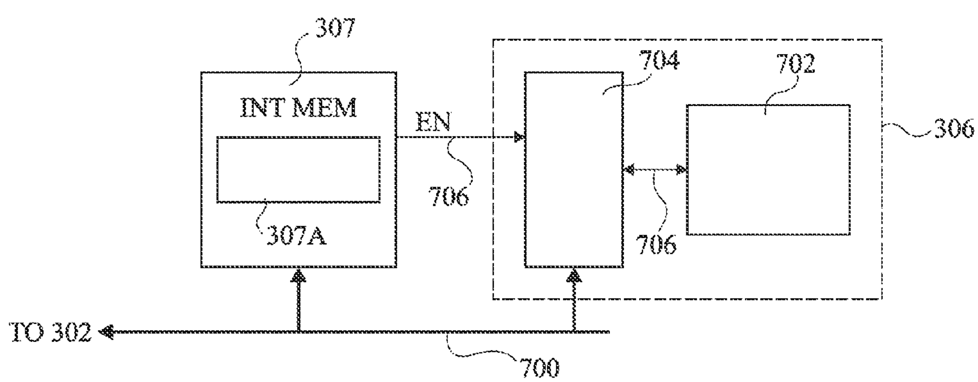


Fig 7

METHOD AND DEVICE FOR FAULT DETECTION

BACKGROUND

1. Technical Field

The present disclosure relates to the field of fault detection, and in particular to a device and method for executing a computing function protected against fault attacks.

2. Description of the Related Art

Integrated circuits may comprise circuitry that is considered sensitive in view of the security of the data that it processes, such as authentication keys, signatures, etc., or of the algorithms it uses, such as encryption or decryption algorithms. Such information should not be communicated to or otherwise be detectable by third parties or unauthorized circuits.

A common process for fraudulently discovering information manipulated by an integrated circuit involves detecting, within the circuit, the zones that are used during the processing of that information. For this, the circuit is activated or otherwise placed in a functional environment, and data to be processed by the circuit is introduced at an input. Then, while the data is processed, for example, the surface of integrated circuit is swept by a laser to inject faults in the functioning of the circuit, and in particular to flip the voltage state stored at one or more nodes of the circuit. While analyzing in parallel the outputs of the circuit, the zones that are used to process the data may be determined. Having localized such zones, the pirate can then concentrate the attacks on these zones in order to discover the secret information.

The injection of faults can also be used to bypass security checks or to infer secret information through the modification of the data being processed.

One solution for protecting against faults attacks is to provide two processing devices arranged to operate in parallel on the same input data. By comparing the results generated by these two devices, the injection of a fault can be detected. However, such a solution comes at a relatively high hardware cost.

An alternative solution that avoids the use of two processing devices is to execute the sensitive function twice using the same processing device and with the same input data. However, a drawback of existing implementations of this type of solution is that they are implemented with relatively high memory resources.

BRIEF SUMMARY

It is an aim of embodiments described herein to at least partially address one or more needs in the prior art.

According to one aspect of the present disclosure, there is provided a method implemented by a processing device comprising: performing a first execution by said processing device of a computing function based on one or more initial parameters stored in a first memory device, the execution of said computing function generating one or more modified values of at least one of said initial parameters, wherein during said first execution said one or more initial parameters are read from said first memory device and said one or more modified values are stored in a second memory device; and performing a second execution by said processing device of said computing function based on said one or more initial parameters stored in said first memory device.

According to one embodiment, during said second execution said one or more initial parameters are read from said first memory device and said one or more modified values are stored in said first memory device.

According to another embodiment, before performing said first execution, the method further comprises storing said one or more initial parameters in said first memory device.

According to another embodiment, the method further comprises, during said first execution of said computing function: receiving by said second memory device a write instruction associated with a first address in said first memory device and with a first data value; storing said first data value in said second memory device and storing said first address as an indexing value in said second memory device in association with said first data value; receiving by said second memory device a read instruction associated with said first address; locating said first data value in said second memory device based on said first address; and outputting said first data value from said second memory device.

According to another embodiment, the method further comprises comparing at least one value generated during said first execution of said computing function with at least one value generated during said second execution of said computing function.

According to another embodiment, said comparing operation comprising reading said at least one value generated during said first execution from said second memory device and reading said at least one value generated during said second execution from said first memory device.

According to another embodiment, the method further comprises: computing a first verification value based on a plurality of values generated by said first execution of said computing function; computing a second verification value based on a plurality of values generated by said second execution of said computing function; comparing said first and second verification values.

According to another embodiment, said first verification value comprises the sum of said plurality of values generated by said first execution and said second verification value comprises the sum of said plurality of values generated by said second execution.

According to another embodiment, said first verification value is computed as a cyclic redundancy check based on said plurality of values generated by said first execution, and said second verification value is computed as a cyclic redundancy check based on said plurality of values generated by said second execution.

According to a further aspect of the present disclosure, there is provided a method of detecting the occurrence of a fault attack during the execution of a computing function, comprising the above method.

According to a further aspect of the present disclosure, there is provided a computing device comprising: a processing device configured to perform first and second executions of a computing function based on one or more initial parameters, said computing function generating one or more modified values of at least one of said initial parameters; a first memory device configured to store said at least one initial parameter; and a second memory device coupled to said processing device and to said first memory device; wherein said processing device is configured to read, during said first execution, said one or more initial parameters from said first memory device and to store, during said first execution, said one or more modified values in said second memory device.

According to one embodiment, said processing device is configured to read, during said second execution, said one or more initial parameters from said first memory device and to store, during said second execution, said one or more modified values in said first memory device.

According to another embodiment, said second memory device comprises an enable input coupled to said processing device, and said second memory device is configured to forward, when disabled, all write and read instructions to said first memory device.

According to another embodiment, said second memory device is configured to receive, during said second execution, read instructions from said processing device and to forward said read instructions to said first memory device if they relate to one of said initial parameters.

According to another embodiment, the computing device further comprises a verification block adapted to compare at least one value generated during said first execution of said computing function with at least one value generated during said second execution of said computing function.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

Non-limiting and non-exhaustive embodiments are described with reference to the following drawings, wherein like labels refer to like parts throughout the various views unless otherwise specified. The sizes and relative positions of elements in the drawings are not necessarily drawn to scale. For example, the shapes of various elements and angles are not drawn to scale, and some of these elements are enlarged and positioned to improve drawing legibility. Further, the particular shapes of the elements as drawn are not intended to convey any information regarding the actual shape of the particular elements and have been solely selected for ease of recognition in the drawings.

The foregoing and other purposes, features, aspects and advantages of embodiments of the present disclosure will become apparent from the following detailed description of embodiments, given by way of illustration and not limitation with reference to the accompanying drawings, in which:

FIG. **1** illustrates a computing device according to an example embodiment;

FIG. **2** is a flow diagram showing operations in a method of executing a computing function according to an example embodiment;

FIG. **3** illustrates a computing device according to an embodiment of the present disclosure;

FIG. **4**A is a flow diagram showing operations in a method according to an embodiment of the present disclosure;

FIG. **4**B is a flow diagram showing operations for implementing a read instruction according to an embodiment of the present disclosure;

FIG. **5** illustrates a computing device according to yet a further embodiment of the present disclosure;

FIG. **6** is a flow diagram showing operations in a method according to a further embodiment of the present disclosure; and

FIG. **7** illustrates a memory interface according to a further embodiment of the present disclosure.

DETAILED DESCRIPTION

Throughout the following description, only those aspects useful for an understanding of the embodiments of the present disclosure will be described in detail. Other aspects, such as the particular computing functions executed by the processing device, have not been described in detail, it being apparent to those skilled in the art that the embodiments described herein are applicable to a broad range of computing functions, for cryptographic applications or other types of applications.

FIG. **1** illustrates an example of a computing device **100** comprising a processing device **102** for executing a computing function, based on instructions stored in an instruction memory **104**. A memory device **106** is coupled to the processing device, and comprises a memory area **106**A providing an execution space used during the execution of the computing function, a memory area **106**B storing a copy of the initial state of memory area **106**A, and a memory area **106**C storing a final state of memory area **106**A.

One or more inputs/outputs **108** may be provided, such as keyboards or keypads, displays, etc.

FIG. **2** is a flow diagram showing operations performed during the execution of the computing function, which is executed twice using the apparatus **100** of FIG. **1**, according to a solution that has been proposed for detecting a fault attack.

In a first operation **201**, the memory **106**, and in particular the memory area **106**A, is initialized. In particular, initial parameters to be used during the execution of the computing function are loaded in the memory area **106**A. These parameters may include certain data values used during the computing function, which could be predetermined values, and/or values received on inputs of the computing device **100**. They could also include one or more cryptographic keys.

In a subsequent operation **202**, the initial state of the execution space **106**A is copied to the memory area **106**B, including the initial parameters.

In a subsequent operation **204**, the computing function is called, which involves loading and executing instructions from the instruction memory **104**, and will result in one or more data values being read from and written to the execution space provided by memory area **106**A.

In a subsequent operation **206**, after the computing function has been executed, the final state present in memory area **106**A is stored in the memory area **106**C.

The computing function is then executed for a second time. As an initial operation **208** of the second execution, the initial state as stored in memory area **106**B is restored in the execution space **106**A.

In a subsequent operation **210**, the computing function **210** is called again, involving loading and executing the instructions again from the instruction memory **104**, and will again result in one or more data values being read from and written to the execution space provided by memory area **106**A.

In a subsequent operation **212**, the new final state present in the memory area **106**A is compared to the final state stored in memory area **106**C, and any difference between the data values of these states would indicate the presence of a fault.

A drawback of the process of FIG. **2** is that, due to the concurrent use of the three memory areas **106**A, **106**B and **106**C, the memory **106** should be of a relatively large size.

FIG. **3** illustrates a computing device **300** according to an embodiment of the present disclosure.

Device **300** comprises a processing device **302**, which is coupled to an instruction memory **304** storing instructions of a computing function to be executed by the processing device **302**. A memory device **306** comprises a memory area **306**A, and an intermediate memory device **307** is coupled between the processing device **302** and the memory device **306**, and comprises a memory area **307**A. In particular, memory device **307** receives address (ADDR) values on lines **308** for read operations, and address and data (DATA) values on line **308** and **310** for write operations, from the processing device **302**. Read data is provided on lines **312** to the processing device **302**. Furthermore, memory device **307** forwards address values for read operations on lines **314**, and address and data values on lines **314** and **316** for write operations. Read data is received by memory device **307** from memory device **306** on lines **318**. Memory device **307** for example comprises an enable input receiving an enable signal EN on line **319** from processing device **302**.

As with the embodiment of FIG. **1**, the computing device **300** may comprise one or more inputs/outputs, labeled **320** in FIG. **3**.

Operation of the computing device **300** will now be described in more detail with reference to the flow diagram of FIG. **4**A. The operations of FIG. **4**A are for example implemented under the control of the processing device **302**.

In a first operation **401**, the memory **306** is initialized. In particular, as described above with reference to operation **201** of FIG. **2**, initial parameters to be used during the execution of the computing function are loaded in the memory area **306**A. These parameters may include certain data values used during the computing function, some of which could be predetermined values, and/or values received by one or more inputs **320** of the computing device **300**. They could also include one or more cryptographic keys. Further examples of the initial parameters include all data forming part of the memory space used by the computing function, such as global variables and local variables.

In a subsequent operation **402**, the computing function is called with the intermediate memory **307** enabled via line **319**. Execution of the computing function involves loading and executing instructions from the instruction memory **304**, and will result in the initial parameters stored in memory area **306**A being read, and one or more modified values of the parameters being generated.

With the intermediate memory **307** enabled, all memory operations originating from the processing device will be first processed by the intermediate memory **307**. Certain write operations, at least those relating to the initial parameters stored in memory area **306**A, are not written to memory area **306**A but are instead written to the memory area **307**A. In particular, the memory **307**A is for example an associative memory. An associative memory is one in which the stored data values are each associated with a further indexing value, and this indexing value is used, during a read operation, to locate the stored data value to be read. Thus, if a write operation of a data value D_1 targets a memory address $ADDR_1$ in memory area **306**A, the data value D_1 is for example written to memory area **307**A, and the address $ADDR_1$ is also stored in memory area **307**A as the indexing value associated with the data value D_1. A future read operation relating to address $ADDR_1$ will be directed to memory area **307**A, and using address $ADDR_1$ as the indexing value, the data value D_1 can be located and read. Thus all read operations relating to data values that have been stored in memory area **307**A will be read from memory device **307**, whereas read operations directed to any of the initial parameters stored in memory **306**A will not be found in memory area **307**A, and will instead be forwarded to the memory area **306**A.

FIG. **4**B illustrates an example of operations performed by the memory device **307**, during the execution of the computing function of operation **402** of FIG. **4**A, in response to a read instruction received from the processing device **302** of FIG. **3**. In a first operation **402**A, a read instruction for an address $ADDR_A$ of memory area **306**A is received by the memory device **307**.

In a subsequent operation **402**B, the memory device **307** is searched, using the address $ADDR_A$ as an indexing value.

In a subsequent operation **402**C, it is determined whether or not the indexing value $ADDR_A$ generated a hit in memory device **307**. If so, the next operation is **402**D, in which the data value associated with this indexing value in memory area **307**A is provided as the output to the processing device **302**. Alternatively, if the indexing value $ADDR_A$ was not found, the next operation is **402**E, in which the read instruction is forwarded to memory device **306**.

Thus the intermediate memory area **307**A provides a memory space in which data may be written and read during the first execution of the computing function, while the initial parameters can be read from the memory area **306**A but are not overwritten.

Referring again to FIG. **4**A, in a subsequent operation **404**, the execution of the computing function is repeated by calling the computing function a second time, but this time with the intermediate memory disabled, for example by a disable signal on line **319**. When disabled, all memory operations received by the intermediate memory device **307** are for example forwarded directly to the memory device **306**. This means that the initial parameters used during the first and second executions of the computing function are the same, and in the absence of faults, the second execution of the computing function should be an identical repetition of the first execution. In one example, the final state stored in the memory area **307**A following the first execution of the computing function is not overwritten during the second execution. The memory area **306**A provides the execution space during the second execution of the computing function, and once this second execution is completed, the memory area **306**A stores the final state.

In a subsequent operation **406**, the results of two executions of the computing function are compared. For example, it is verified that the data value stored at each address in the memory area **307**A is identical to the data value of the corresponding address of the memory area **306**A. Discrepancies between corresponding data values in the memory areas **306**A, **307**A could indicate the injection of a fault in one of these memories, in the processing device, or in the instruction memory **304**. If the verification indicates the presence of a fault, a countermeasure may be taken, such as resetting the processing device, erasing the memory areas **306**A, **307**A, and/or incrementing a count value leading to a permanent deactivation of the processing device if a certain number of faults is detected.

FIG. **5** illustrates a computing device **500** according to an alternative embodiment. Device **500** comprises many elements in common with computing device **300** of FIG. **3**, and those elements have been labeled with like numerals and will not be described again in detail.

Device **500** comprises a verification block **502** coupled to receive the address and data values transmitted on lines **308**, **310** and **312** between the processing device **302** and the intermediate memory device **307**. Furthermore, the verification block **502** may also receive instruction data from the instruction memory **304** on a line **504**, which is coupled to the connection between the instruction memory **304** and the processing device **302**. Thus the verification block **502** for example receives a copy of all instructions loaded to the processing device **302** during the execution of the computing functions. The verification block **502** records the data from these various sources by calculating a checksum value, for example equal to the sum of all of the values it receives. For example, assuming that the data values are n-bit values, the sum could be calculated as the sum, modulo n, of the data values and address values. The value of n could for example be between **8** and **64** bits. The verification block **502** for example calculates a first checksum during the first execution of the computing function, and a second checksum during the second execution of the computing function, and compares these checksums to verify that they match. If they do not match, this would imply the injection of a fault in one of the memories **304**, **306**, **307** or into the processing device **302** during the first or second execution.

A modified operation flow based on the use of the verification block **502** will now be described with reference to the flow diagram of FIG. **6**. The operations of FIG. **6** are for example implemented under the control of the processing device **302**.

In a first operation **601**, the memory **306** is initialized. In particular, as described above with reference to operation **401** of FIG. **4**A, initial parameters to be used during the execution of the computing function are loaded in the memory area **306**A.

Then, in a subsequent step **602**, the checksum implemented by the verification block **502** is started, such that from this moment on, all the data signals and the address signals provided on the lines **308** to **312**, and optionally the instructions from the instruction memory **304**, are summed.

In a subsequent operation **604**, in a similar manner to operation **402** of FIG. **4**A, the computing function is called with the intermediate memory **307** enabled. Thus the memory area **307**A is used as the execution space, and only read operations relating to the initial parameters are forwarded to the memory device **306**.

In subsequent operation **606**, the calculation of the checksum by the verification block **502** is stopped, and the value reached is for example stored for future verification.

In a subsequent operation **608**, the checksum is activated again in preparation for the second execution of the computing function.

In a subsequent operation **610**, the execution of the computing function is repeated by calling it a second time. As with operation **404** of FIG. **4**A, the intermediate memory device **307** is disabled during the second execution during operation **610**. Thus, during the second execution, all memory operations received by the intermediate memory device **307** are for example forwarded directly to the memory device **306**.

The calculation by the verification block **502** of the checksum during the second execution of the computing function could be implemented in a number of different ways. One option, assuming that the first checksum calculated during the first execution has been transferred to a separate register/memory, would be to simply reset the register used to accumulate the checksum, and to calculate the second checksum in this register. Alternatively, the verification block **502** may comprise two registers, a first of which is used to accumulate and store the checksum during the first execution of the computing function, and a second of which is used to accumulate and store the checksum during the second execution of the computing function. As a further option, the first checksum could be calculated by an addition of all the data/address/instruction values received, and the result could remain in the same register after the first execution of the computing function. Then, during the second execution of the computing function, the data values received by the verification block **502** could be subtracted from the first checksum such that the value in the checksum register would equal zero by the end of the second execution if no faults are present.

A subsequent operation **612** of FIG. **6** involves stopping the checksum and verifying the value, for example by comparing the first and second checksums and verifying that they are equal. As previously, if the verification of the checksum indicates the presence of a fault, a countermeasure may be taken, such as resetting the processing device, erasing the memory devices **306**, **307** and/or incrementing a count value leading to a permanent deactivation of the processing device **302** if a certain number of faults is detected.

The memory devices **306**, **307** could each be implemented by a RAM (Random Access Memory), such as an SRAM (Static RAM) or other type of volatile programmable memory device. Alternatively, the memory device **306** could be implemented by a non-volatile memory, such as for example an E^2PROM (electronically erasable programmable read-only memory), as will now be described with reference to FIG. **7**.

FIG. **7** illustrates the memory device **306** and the intermediate memory device **307**, each of which are coupled to a bus **700**, which is also for example coupled to the processing device **302** (not shown in FIG. **7**). Memory device **306** is a non-volatile memory comprising a non-volatile memory array **702**, which receives control and data signals for read and write operations from a memory interface module **704** on control lines **706**. The memory interface module **704** is coupled to the bus **700** for receiving the address and data signals corresponding to write and read operations, which it converts to a suitable format for the non-volatile memory array **702**. An enable signal is for example provided on a line **706** from the intermediate memory device **307** to the memory interface **704**, allowing the intermediate memory device **307** to control the activation of the non-volatile memory device **306**.

During the first execution of the computing function, for example corresponding to operations **402** and **604** described above, the enable line **706** for example deactivates the non-volatile memory **306**, and the intermediate memory **307** performs all read and write operations, unless they concern read operations of the initial parameters. For example, if the intermediate memory **307** receives a read operation request for an initial parameter not stored in its memory area **307**A, it activates the memory interface module via line **706** so that the read operation is processed by the non-volatile memory **704** and the corresponding data is read from the array **702**.

During the second execution of the computing function, for example corresponding to operations **404** and **610** described above, the enable line **706** for example activates the non-volatile memory **306**, and the non-volatile memory performs all read and write operations.

An advantage of the embodiments described herein is that the execution of a computing function may be repeated with the use of relatively low memory resources. In particular, the initial parameters are stored only once during the first execution of the computing function, thereby economizing memory space. Furthermore, the use of a checksum avoids saving the entire final state generated during the first execution of the computing function.

Having thus described a number of embodiments, various alterations, modifications and improvements will readily occur to those skilled in the art.

For example, it will be apparent to those skilled in the art that the particular hardware implementation of the embodiments described herein will depend of the particular application, and could include separate memory devices or a single memory device containing the memory areas **306**A, **307**A and the instruction memory **304**. Furthermore, the particular control of the memory device(s) during read and write operations will depend on the types of memory used.

It will also be apparent to those skilled in the art that various different checksum algorithms could be used to compute the checksums, a simple addition of the values being just one example. Furthermore, the checksum could be implemented by a cyclic redundancy check.

The operations of the various flow diagrams of FIG. **4**A, **4**B and **6** may be implemented entirely by software executed by the processing device **302** of FIG. **3**, although certain operations, such as enabling or disabling the memory devices **306** and/or **307** and comparing results/checksums may be implemented by a state machine, for example forming part of the processing device **302**.

Furthermore, the various features described in relation to the various embodiments could, in alternative embodiments, be combined in any combination.

The various embodiments described above can be combined to provide further embodiments. These and other changes can be made to the embodiments in light of the above-detailed description. In general, in the following claims, the terms used should not be construed to limit the claims to the specific embodiments disclosed in the specification and the claims, but should be construed to include all possible embodiments along with the full scope of equivalents to which such claims are entitled. Accordingly, the claims are not limited by the disclosure.

The invention claimed is:

1. A method implemented by a processing device, comprising:

performing a first execution by said processing device of a computing function based on one or more initial parameters stored in a first memory device, the first execution of said computing function generating one or more modified values of at least one of said initial parameters, wherein during said first execution said one or more initial parameters are read from said first memory device and said one or more modified values are stored in a second memory device;

performing a second execution by said processing device of said computing function based on said one or more initial parameters stored in said first memory device, the second execution of said computing function generating one or more second modified values; and

detecting an occurrence of a fault attack based on a difference between the one or more modified values generated during the first execution and the one or more second modified values generated during the second execution, wherein said one or more initial parameters are read from said first memory device during said first execution after searching said second memory device for modified values of said one or more initial parameters.

2. The method of claim **1**, wherein said one or more modified values are stored in said second memory device in response to a write operation targeting a memory address of said one or more initial parameters in said first memory device.

3. The method of claim **1** wherein during said second execution said one or more initial parameters are read from said first memory device and said one or more second modified values are stored in said first memory device.

4. The method of claim **1**, further comprising:

before performing said first execution, storing said one or more initial parameters in said first memory device.

5. The method of claim **1**, further comprising, during said first execution of said computing function:

receiving by said second memory device a write instruction associated with a first address in said first memory device and with a first data value;

storing said first data value in said second memory device and storing said first address as an indexing value in said second memory device in association with said first data value;

receiving by said second memory device a read instruction associated with said first address;

locating said first data value in said second memory device based on said first address; and

outputting said first data value from said second memory device.

6. The method of claim **1**, further comprising:

comparing at least one value generated during said first execution of said computing function with at least one value generated during said second execution of said computing function.

7. The method of claim **6** wherein said comparing operation comprises reading said at least one value generated during said first execution from said second memory device and reading said at least one value generated during said second execution from said first memory device.

8. The method of claim **1**, further comprising:

computing a first verification value based on a plurality of values generated by said first execution of said computing function;

computing a second verification value based on a plurality of values generated by said second execution of said computing function; and

comparing said first and second verification values.

9. The method of claim **8** wherein said first verification value comprises a sum of said plurality of values generated by said first execution and said second verification value comprises a sum of said plurality of values generated by said second execution.

10. The method of claim **8** wherein said first verification value is computed as a cyclic redundancy check based on said plurality of values generated by said first execution, and said second verification value is computed as a cyclic redundancy check based on said plurality of values generated by said second execution.

11. A computing device comprising:

a processing device configured to perform first and second executions of a computing function based on one or more initial parameters, said computing function generating one or more modified values of at least one of said initial parameters;

a first memory device configured to store said at least one initial parameter; and

a second memory device coupled to said processing device and to said first memory device;

wherein said processing device is configured to search, during said first execution, said second memory device for modified values of said initial parameters, read, during said first execution, said one or more initial parameters from said first memory device, and to store, during said first execution, said one or more modified values in said second memory device, and wherein said processing device is configured to store said one or more modified values in said second memory device in response to a write operation targeting a memory address of said one or more initial parameters in said first memory device, wherein said processing device is configured to detect an occurrence of a fault attack based on a difference between one or more modified values generated during the first execution and one or more second modified values generated during the second execution.

12. The computing device of claim **11** wherein, before reading said one or more initial parameters from said first memory device during said first execution, said processing device is configured to search said second memory device for modified values of said one or more initial parameters.

13. The computing device of claim **11** wherein said processing device is configured to read, during said second execution, said one or more initial parameters from said first memory device and to store, during said second execution, said one or more modified values in said first memory device.

14. The computing device of claim 11 wherein said second memory device comprises an enable input coupled to said processing device, and wherein said second memory device is configured to forward, when disabled, all write and read instructions to said first memory device.

15. The computing device of claim 11 wherein said second memory device is configured to receive, during said second execution, read instructions from said processing device and to forward said read instructions to said first memory device if they relate to one of said initial parameters.

16. The computing device of claim 11, further comprising:
a verification block adapted to compare at least one value generated during said first execution of said computing function with at least one value generated during said second execution of said computing function.

17. A memory configured to store computing instructions, the computing instructions configured to direct a processor in a computing device to perform a method of detecting an occurrence of a fault attack, the method comprising:
storing at least one initial parameter in a first memory device;
reading the at least one initial parameter from the first memory device;
executing a computing function a first time, the computing function arranged to direct execution based on the at least one initial parameter;
generating in the computing function at least one modified initial parameter;
storing the at least one modified initial parameter in a second memory device;
executing the computing function a second time, the computing function arranged to direct execution based on the at least one initial parameter;
generating in the computing function at least one second modified initial parameter; and
detecting an occurrence of a fault attack based on a difference between the at least one modified initial parameter generated during the first execution of the computing function and the at least one second modified initial parameter generated during the second execution of the computing function, wherein the at least one initial parameter is read from the first memory device during the first execution of the computing function after the second memory device is searched for at least one modified value of the at least one initial parameter.

18. The memory of claim 17 wherein the computing instructions are configured to direct the processor to perform the method, the method further comprising:
receiving by the second memory device a first instruction associated with a first address in the first memory device and with a first data value;
storing the first data value in the second memory device;
storing the first address as an indexing value in the second memory device in association with the first data value;
receiving by the second memory device a second instruction associated with the first address;
locating the first data value in the second memory device based on the first address; and
outputting the first data value from the second memory device.

19. The memory of claim 17 wherein detecting the occurrence of the fault attack includes combining a plurality of modified initial parameters using a certain algorithm and combining a plurality of second modified initial parameters using the certain algorithm.

20. The memory of claim 19 wherein the certain algorithm is a summation algorithm or a cyclic redundancy algorithm.

21. The memory of claim 17 wherein storing the at least one initial parameter in the first memory device and storing the at least one modified initial parameter in the second memory device comprise storing the parameters in a single non-volatile memory device.

22. The memory of claim 17 wherein the computing instructions are configured to direct the processor to perform the method, the method further comprising:
asserting a fault indication signal when an occurrence of a fault attack is detected.

* * * * *